Eric Roberts
CS 106A

Handout #31A
April 30, 2012

Solution to Section #4

1. Scrabble scoring

```
/*
 * File: ScrabbleScore.java
 * ------------------------
 * This program reads in words and calculates their score in
 * Scrabble.  Only uppercase letters count, so that it is possible
 * to use lowercase letters to indicate blanks, so that the input
 * "QUIZzICAL" shows that the second Z is actually a blank.
 */

import acm.program.*;

public class ScrabbleScore extends ConsoleProgram {

   public void run() {
      println("Program to compute Scrabble scores");
      while (true) {
         String word = readLine("Enter a word: ");
         if (word.length() == 0) break;
         println(word + " scores " + scoreScrabbleWord(word));
      }
   }

/**
 * Computes the score for a Scrabble word.
 *
 * @param word The Scrabble word, using lowercase letters to represent blanks
 * @return The score for that word
 */
   private int scoreScrabbleWord(String word) {
      int score = 0;
      for (int i = 0; i < word.length(); i++) {
         score += scoreScrabbleLetter(word.charAt(i));
      }
      return score;
   }

/* Computes the score for a Scrabble letter */
   private int scoreScrabbleLetter(char ch) {
      if (!Character.isUpperCase(ch)) return 0;
      switch (ch) {
        case 'Q': case 'Z':
         return 10;
        case 'J': case 'X':
         return 8;
        case 'K':
         return 5;
        case 'F': case 'H': case 'V': case 'W': case 'Y':
         return 4;
        case 'B': case 'C': case 'M': case 'P':
         return 3;
        case 'D': case 'G':
         return 2;
        default:
         return 1;
      }
   }
}
```

2. StanfordSpeak

```
/*
 * File: StanfordSpeak.java
 * ------------------------
 * This file tests the createStanfordAbbreviation method, which turns
 * "Coffee House" into "CoHo", "Memorial Church" into "MemChu",
 * "Frozen Yogurt" into "FroYo", "Sophomore College" into "SoCo",
 * and so forth.
 */

import acm.program.*;
import java.util.*;

public class StanfordSpeak extends ConsoleProgram {

   public void run() {
      println("Enter complete phrases (use blank line to stop).");
      while (true) {
         String line = readLine("Phrase: ");
         if (line.length() == 0) break;
         println(createStanfordAbbreviation(line));
      }
   }

/**
 * Returns the StanfordSpeak abbreviation for the specified string.  This
 * abbreviation consists of all characters from each word up to the first
 * vowel, along with the next letter if that vowel is followed by m or n.
 * Everything except whitespace is considered a valid word constituent.
 * If no vowels appear in a word, the abbreviation includes the entire word.
 *
 * @param str The multiword string from which the abbreviation is constructed
 * @return The StanfordSpeak abbreviation
 */
   private String createStanfordAbbreviation(String str) {
      String result = "";
      StringTokenizer tokenizer = new StringTokenizer(str);
      while (tokenizer.hasMoreTokens()) {
         String token = tokenizer.nextToken();
         result += abbreviateWord(token);
      }
      return result;
   }

/* Returns the initial substring used to form a StanfordSpeak abbreviation */
   private String abbreviateWord(String word) {
      int vp = findFirstVowel(word);
      if (vp == -1) return word;
      String result = word.substring(0, vp + 1);
      if (vp + 1 < word.length() && isSonorant(word.charAt(vp + 1))) {
         result += word.charAt(vp + 1);
      }
      return result;
   }
```

☞

```
/* Returns true if the character is an m or an n */
   private boolean isSonorant(char ch) {
      ch = Character.toLowerCase(ch);
      return ch == 'm' || ch == 'n';
   }

/* Returns the index of the first vowel in the word (-1 if none) */
   private int findFirstVowel(String word) {
      for (int i = 0; i < word.length(); i++) {
         if (isEnglishVowel(word.charAt(i))) return i;
      }
      return -1;
   }

/* Returns true if the character is a vowel */
   private boolean isEnglishVowel(char ch) {
      switch (Character.toLowerCase(ch)) {
         case 'a': case 'e': case 'i': case 'o': case 'u':
            return true;
         default:
            return false;
      }
   }

}
```

3. Removing characters from a string

```
/**
 * Removes all occurrences of ch from str.
 *
 * @param str The string from which you are removing characters
 * @param ch The character you are removing
 * @return The original string with all copies of ch removed
 */
   private String removeAllOccurrences(String str, char ch) {
      String result = "";
      for (int i = 0; i < str.length(); i++) {
         if (str.charAt(i) != ch) result += str.charAt(i);
      }
      return result;
   }
```

4. Adding commas to numeric strings

```
/**
 * Adds commas to a numeric string to separate groups of three digits.
 *
 * @param digits The digit string without commas
 * @result A string in which commas separate the digits into groups of three
 */
   private String addCommasToNumericString(String digits) {
      String result = "";
      int len = digits.length();
      int nDigits = 0;
      for (int i = len - 1; i >= 0; i--) {
         result = digits.charAt(i) + result;
         nDigits++;
         if (nDigits % 3 == 0 && i > 0) {
            result = "," + result;
         }
      }
      return result;
   }
```